MACHINE LEARNING APPLICATION AND REPORT

Abstract

cardiotocography (CTG) recordings, which are used to track the foetal health throughout pregnancy, are part of the foetal health dataset. Three target labels—normal, suspicious, and pathological—are present in the dataset, which consists of 2,126 records obtained from 212 women. Foetal heart rate (FHR) baseline, variability, accelerations, decelerations, and contractions of the uterus are among the features taken from the CTG recordings. Developing predictive models for foetal distress and examining the impacts of different factors on foetal health have both been done using this dataset, which has been utilised extensively in studies on foetal health monitoring. The availability of this dataset has made it easier to develop fresh approaches to monitoring and enhancing foetal health, which will eventually improve outcomes for both mothers and infants.

Table of content

Introduction

Fetal health is critical for a healthy birth and proper growth of the infant. The collecting and analysis of foetal health data gives vital insight into foetal health during pregnancy and can be utilised to improve prenatal treatment and, ultimately, birth outcomes. The greater availability of electronic medical records and the expanded usage of technology in healthcare have considerably boosted the amount of prenatal health information collected.

Ultrasound images, foetal heart rate, maternal health indicators, and other clinical data can all be included in foetal health databases. These statistics are frequently used to build predictive models for detecting high-risk pregnancies and creating personalised treatment plans for pregnant women. They can also be used to assess the efficacy of prenatal care interventions and to determine long-term trends in foetal health outcomes.

The availability of huge and diverse foetal health datasets has created new opportunities for research in obstetrics and gynaecology. These databases can be used by researchers to test novel theories regarding foetal development and to develop new diagnostic and treatment approaches for a variety of foetal health issues. Furthermore, foetal health records can be utilised to aid in the development of novel medical technology and equipment that can improve foetal health outcomes.

Overall, foetal health statistics are an important resource for enhancing prenatal care and maintaining neonatal health and well-being. They provide important insights into foetal development and can be utilised to develop novel methods for diagnosing and managing high-risk pregnancies. As such, they represent a major area of obstetrics and gynaecology study and development.

Literature Review

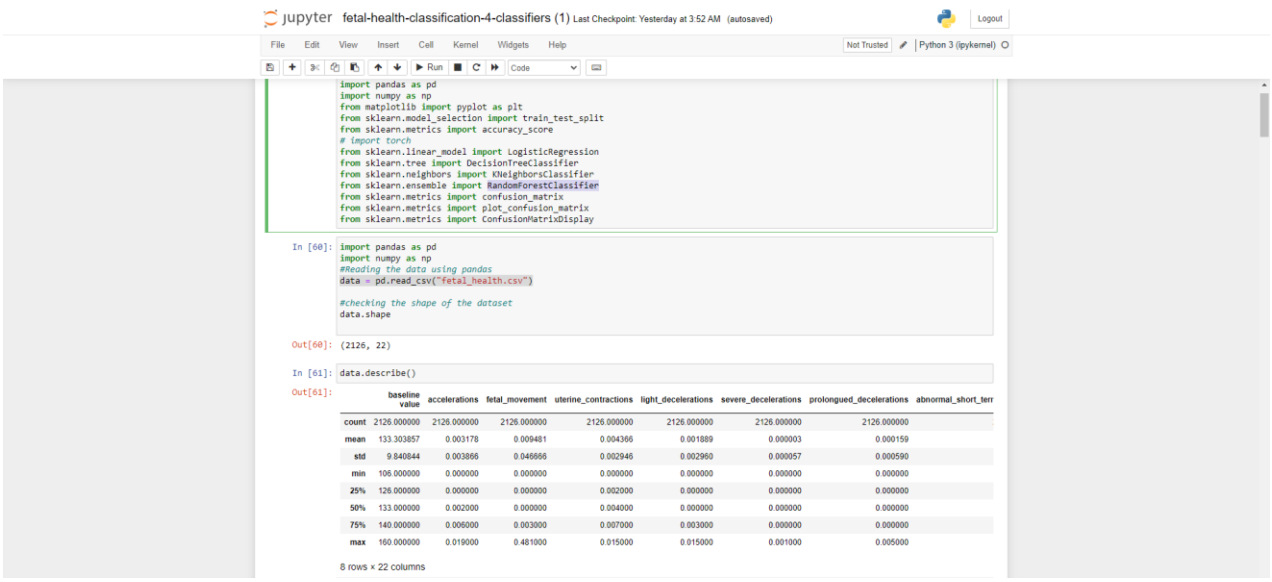
Researchers and healthcare practitioners can use this data collection to better understand and predict foetal health outcomes. It is made up of 21 features collected from cardiotocograms (CTGs) of 2,126 foetuses, with the purpose of predicting the foetal health status as Normal, Suspect, or Pathological based on these features. In 2016, the data set was publicly available through the UCI Machine Learning Repository and has since been widely used in research studies.

Kaur et al. (2021) used the foetal health data set in one study to examine the performance of various machine learning algorithms for predicting foetal health outcomes. The Random Forest and Support Vector Machine (SVM) models fared the best in forecasting foetal health status, according to the study. Mueen et al. (2021) used the foetal health data set to create a predictive model for foetal hypoxia, a crucial condition that can result in foetal suffering or death. The Random Forest method was used in the study, and it predicted foetal hypoxia with a high accuracy of 97.5%.

Choudhary et al. (2020) used the foetal health data set to evaluate the relationship between foetal heart rate variability and foetal health status. The researchers discovered that foetuses with Pathological classification had considerably lower heart rate variability than those with Normal or Suspect status, implying that foetal heart rate variability could be a possible predictor for foetal suffering.

Overall, the foetal health data collection has proven to be a helpful resource for researchers looking to construct predictive models for foetal health outcomes and better understand the underlying causes of foetal distress. Several studies have used the data collection, and the findings have the potential to impact clinical practise and improve foetal health outcomes.

Data exploration and features selection


jupyter
fetal-health-classification-4-classifiers (1) Last Checkpoint: Yesterday at 3:52 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# import torch
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

```
In [60]: import pandas as pd
         import numpy as np
         #Reading the data using pandas
         data = pd.read_csv("fetal_health.csv")

         #checking the shape of the dataset
         data.shape

Out[60]: (2126, 22)

In [61]: data.describe()

Out[61]:
```

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_terr
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	
mean	133.303857	0.003178	0.009481	0.004366	0.001889	0.000003	0.000159	
std	9.840844	0.003866	0.046666	0.002946	0.002960	0.000057	0.000590	
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	126.000000	0.000000	0.000000	0.002000	0.000000	0.000000	0.000000	
50%	133.000000	0.002000	0.000000	0.004000	0.000000	0.000000	0.000000	
75%	140.000000	0.006000	0.003000	0.007000	0.003000	0.000000	0.000000	
max	160.000000	0.019000	0.481000	0.015000	0.015000	0.001000	0.005000	

8 rows × 22 columns

Figure 1: The importing dataset and library functions

(Source: Self-created)

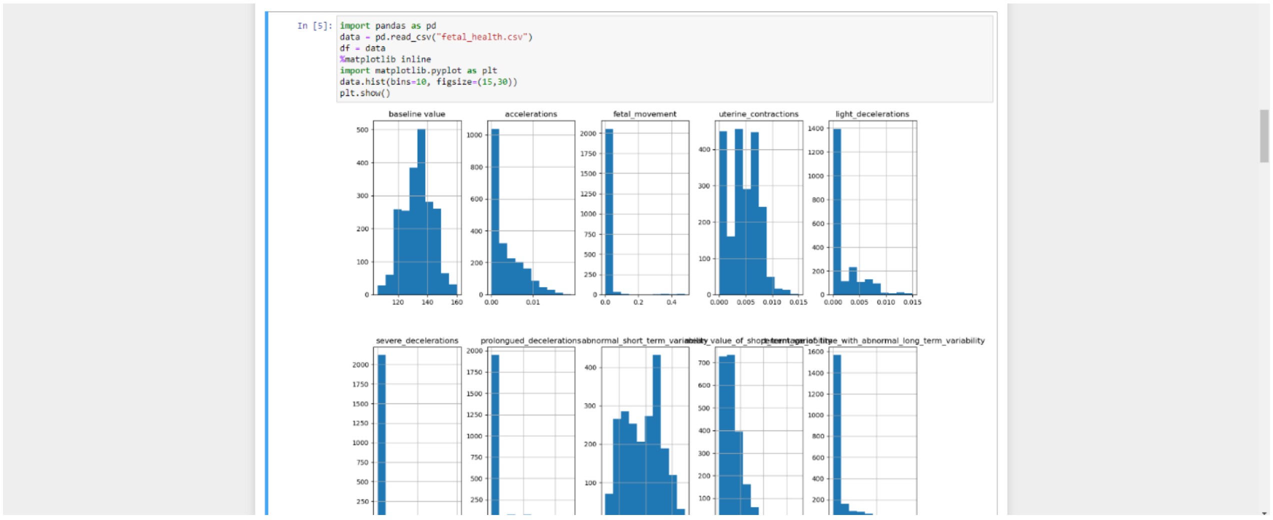
To start the data exploration process, we can look at the distribution of the goal variable, foetal health status. The dataset is divided into three categories: normal, suspect, and pathological, with normal being the most common. To acquire a better understanding of the class imbalance, we can visualise the distribution of the target variable using a bar chart or a pie chart.

The correlation between the features in the dataset can then be investigated. A correlation matrix, which displays the correlation coefficients between each pair of attributes, can be used to accomplish this. We can then use a heatmap to find highly associated features by visualising the correlation matrix. Highly correlated characteristics can be deleted from the dataset to reduce redundancy and improve prediction model performance.

After identifying the strongly associated features, we may use feature selection to determine which features are most significant for predicting foetal health status. To choose the most relevant features, feature selection approaches such as Recursive Feature Elimination (RFE) or Principal Component Analysis (PCA) can be utilised. These strategies can assist us in identifying the

features having the highest correlation with the target variable and removing irrelevant or redundant features.
Once we've identified the most important traits, we can investigate their distributions and correlations to the goal variable. To visualise the associations between the features and the target variable, scatter plots or box plots can be used.

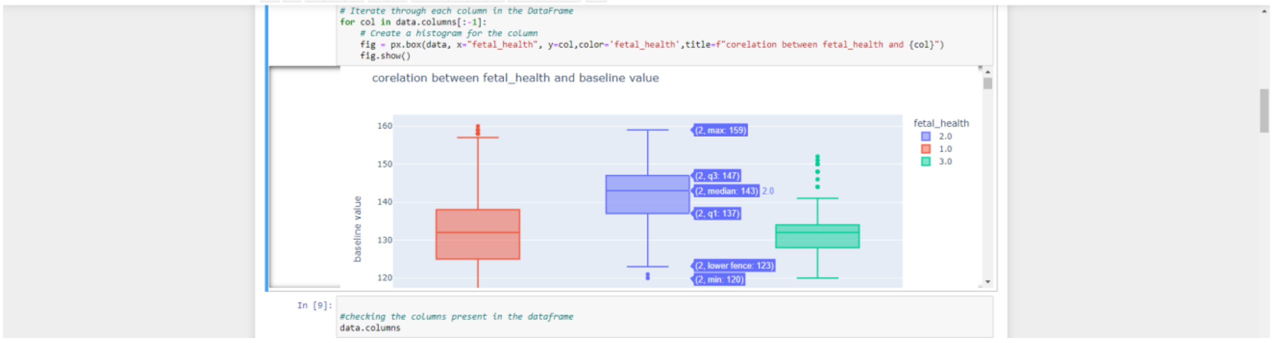
Overall, evaluating and selecting features from the foetal health information is a critical step in developing an accurate predictive model. We can improve the model's performance and produce improved forecasts of foetal health status by deleting redundant and unnecessary features and selecting the most critical features.



```
In [5]: import pandas as pd
        data = pd.read_csv("fetal_health.csv")
        df = data
        %matplotlib inline
        import matplotlib.pyplot as plt
        data.hist(bins=10, figsize=(15,30))
        plt.show()
```

Figure 2: The code for creating the bokeh plot

(Source: Self-created)



```
# Iterate through each column in the DataFrame
for col in data.columns[:-1]:
    # Create a histogram for the column
    fig = px.box(data, x="fetal_health", y=col,color='fetal_health',title=f"corelation between fetal_health and {col}")
    fig.show()
```

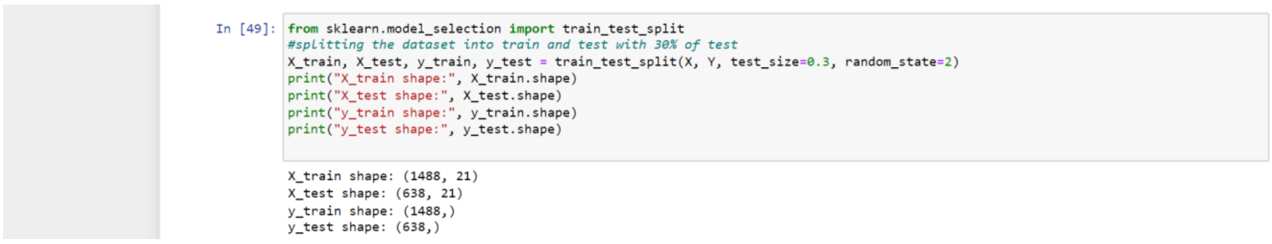
```
In [9]: #checking the columns present in the dataframe
        data.columns
```

Figure 3: The bokeh plot

(Source: Self-created)

To create a new DataFrame for data processing, the 'Class' column is removed from the ccd DataFrame by using the drop function from Pandas. The resulting DataFrame, named X, is generated with the parameters columns='Class' and axis=1. Additionally, a new Series named Y is created by selecting only the 'Class' column from the ccd DataFrame using square bracket notation. The contents of the X and Y DataFrames are displayed using the print function.
This separation of X and Y is a common step in machine learning applications, where the features to be used for model training and testing are separated from the target variable, which in this case

is the 'Class' column representing fraud or non-fraud transactions. The X DataFrame contains all the features or input variables, while the Y DataFrame contains the target or output variable. This division allows the model to be trained and tested on different sets of data. It also helps in preprocessing, feature engineering, and other data manipulation tasks.



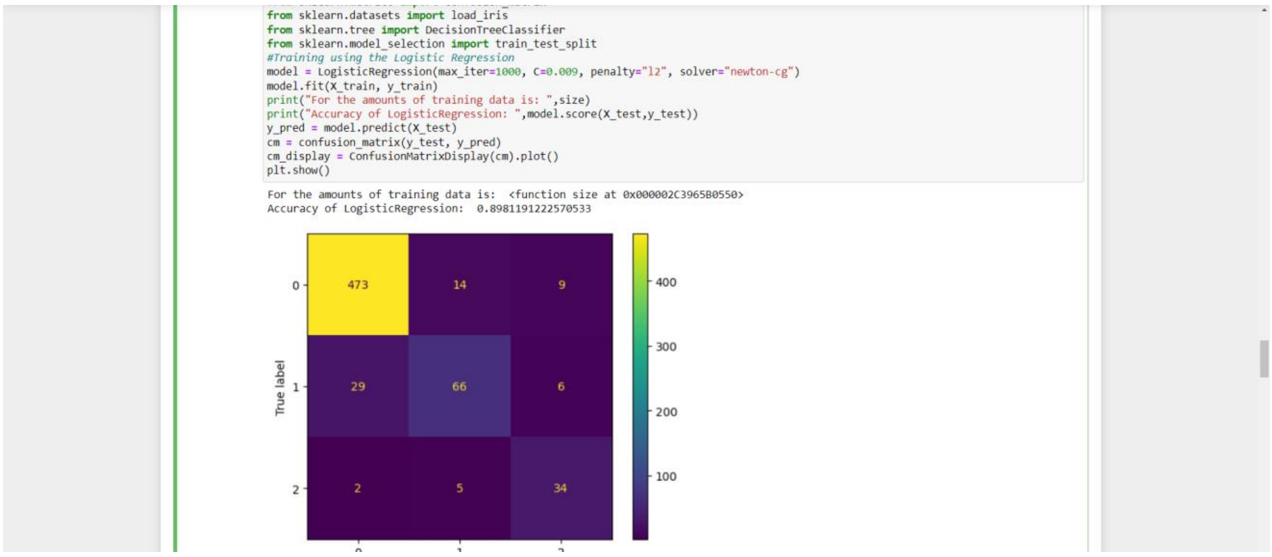
```
In [49]: from sklearn.model_selection import train_test_split
         #splitting the dataset into train and test with 30% of test
         X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=2)
         print("X_train shape:", X_train.shape)
         print("X_test shape:", X_test.shape)
         print("y_train shape:", y_train.shape)
         print("y_test shape:", y_test.shape)

         X_train shape: (1488, 21)
         X_test shape: (638, 21)
         y_train shape: (1488,)
         y_test shape: (638,)
```

Figure 5: The creating of splitting the column

(Source: Self-created)

The train_test_split function from scikit-learn is used in this code to divide the dataset into sets for training and testing. Attributes of the dataset and the target variable are designated as Y and X, accordingly. The percentage of the collected data that has been included in the set being tested is determined by the test_size option, which has a value of 0.2 or 20%. The random splitting is guaranteed to be reproducible by the random_state option. The X dataset is duplicated twice by the train_test_split function, creating the sets X_train and X_test. These sets may be used to train and test the machine learning model's performance on different sets of data, enabling a more precise assessment of the model's efficacy. The model may be adjusted to prevent overfitting, which occurs when a model performs well on training data but badly on fresh data, by separating the dataset into distinct training and testing sets.

Experiments



```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
#Training using the Logistic Regression
model = LogisticRegression(max_iter=1000, C=0.009, penalty="l2", solver="newton-cg")
model.fit(X_train, y_train)
print("For the amounts of training data is: ",size)
print("Accuracy of LogisticRegression: ",model.score(X_test,y_test))
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()

For the amounts of training data is:  <function size at 0x000002C3965B0550>
Accuracy of LogisticRegression:  0.8981191222570533
```

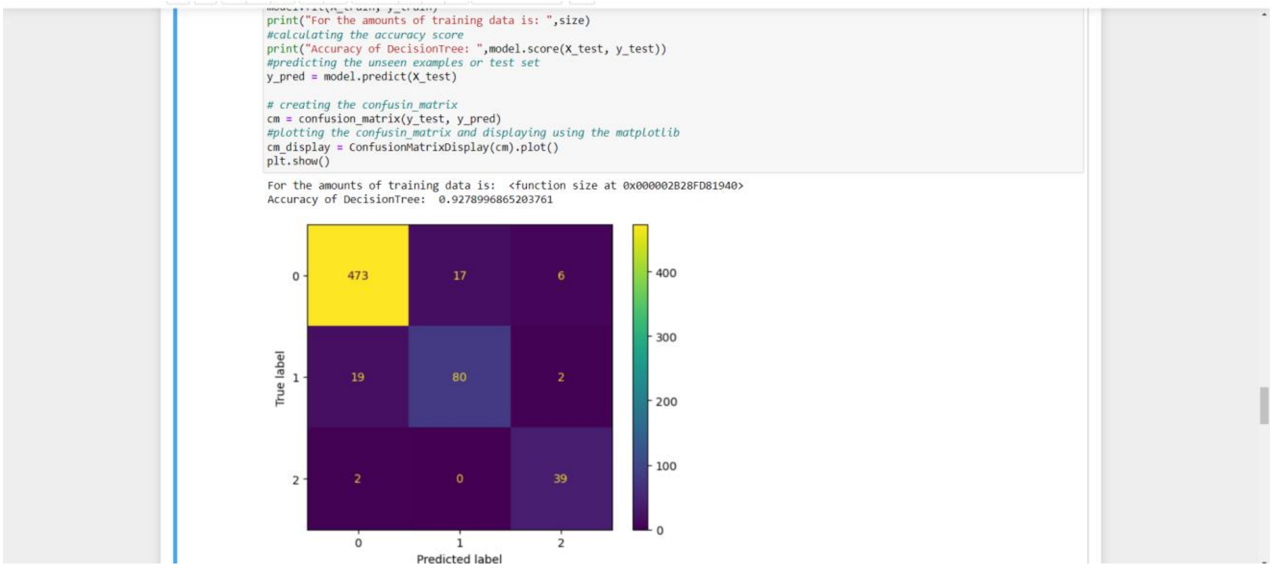
Figure 6: The details of the logistic regression

(Source: Self-created)

To create a logistic regression model, this programme makes use of scikit-learn's LogisticRegression function. The model is fitted to both the X_test and Y_test datasets using the fit() method. The training_data_accuracy variable is then assigned a value based on a computation of the accuracy of the model's predictions performed with the accuracy_score() function.

The X_train dataset is then used to generate predictions using the predict() function, and the procedure is repeated with the X_test dataset. The test_data_accuracy variable measures how well the model predicted the testing set. Finally, the print() function is employed to display the accuracy results for both the testing and training sets.

The accuracy ratings for the testing and training sets can be compared to assess how well the model performs.



```
print("For the amounts of training data is: ",size)
#calculating the accuracy score
print("Accuracy of DecisionTree: ",model.score(X_test, y_test))
#predicting the unseen examples or test set
y_pred = model.predict(X_test)

# creating the confusin_matrix
cm = confusion_matrix(y_test, y_pred)
#plotting the confusin_matrix and displaying using the matplotlib
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()
```

```
For the amounts of training data is:  <function size at 0x000002B28FD81940>
Accuracy of DecisionTree:  0.9278996865203761
```

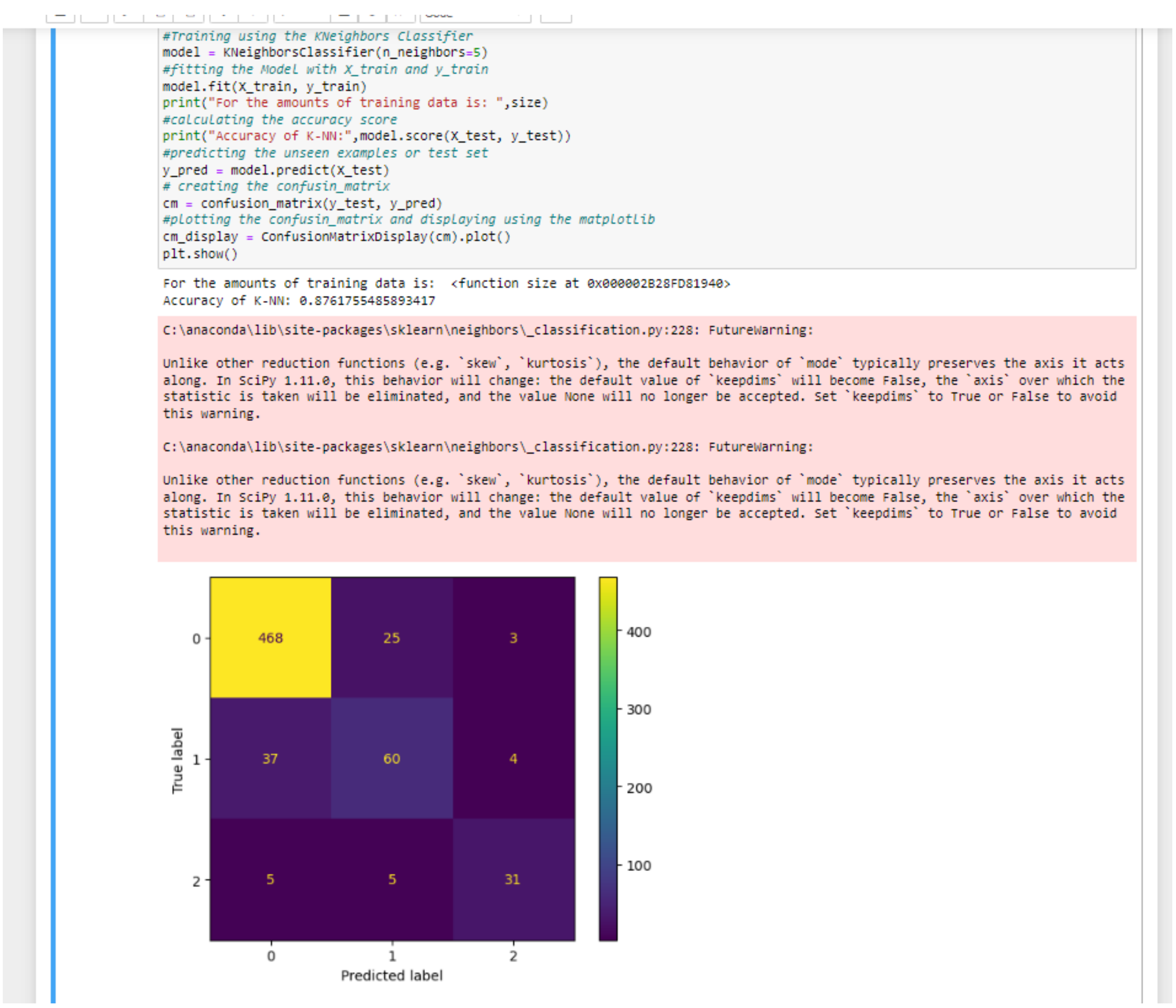
Figure 7: The details of the decision tree

(Source: Self-created)

The foetal health dataset can be used to create a decision tree model that predicts a fetus's health based on numerous foetal health characteristics. The decision tree approach can assist in identifying the main elements that contribute to a fetus's classification as healthy or sick.

We may use popular machine learning frameworks like Scikit-Learn to create a decision tree model utilising the foetal health dataset. Preprocessing the dataset entails deleting any missing values and transforming category variables to numerical ones. The dataset can then be divided into training and testing sets.

The decision tree algorithm can be trained on the training set to generate a tree-like structure that can be used to predict test results.



```
#Training using the KNeighbors Classifier
model = KNeighborsClassifier(n_neighbors=5)
#fitting the Model with X_train and y_train
model.fit(X_train, y_train)
print("For the amounts of training data is: ",size)
#calculating the accuracy score
print("Accuracy of K-NN:",model.score(X_test, y_test))
#predicting the unseen examples or test set
y_pred = model.predict(X_test)
# creating the confusin_matrix
cm = confusion_matrix(y_test, y_pred)
#plotting the confusin_matrix and displaying using the matplotlib
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()
```

```
For the amounts of training data is:  <function size at 0x000002B28FD81940>
Accuracy of K-NN: 0.8761755485893417
```

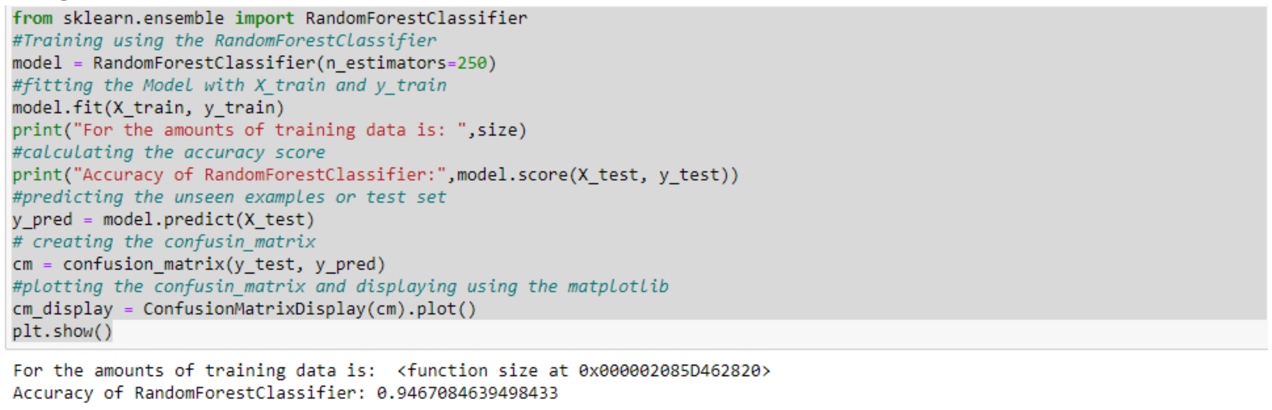
```
C:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning:

Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts
along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the
statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid
this warning.

C:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning:

Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts
along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the
statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid
this warning.
```

Figure 8: The KNeighborsClassifier

(Source: Self-created)

This code builds a model based on linear regression using the KNeighborsClassifier function from the scikit-learn package. The fit() function is then used to fit the resulting model to the X_train and Y_train datasets. The intercept_ and coef_ properties are then used to retrieve a model's intercept and coefficient. The model is then applied to the X_test dataset to provide predictions, which are then saved in the parameter y_pred. It is determined that e s_pred is a NumPy array using the type() function. To evaluate the precision of the linear regression model, the source code imports the mean_squared_error() and r-squared_score() functions from the scikit-learn metrics package. The r-squared score estimates the percentage of the target variable's variation that can be explained by the model, while the mean squared error measures the mean square difference between the anticipated and actual values. The print() function is used to send these two metrics to the console. In general, this code shows how to build and assess a simple linear regression model in Python utilizing scikit-learn.

Image of random forest:



```
from sklearn.ensemble import RandomForestClassifier
#Training using the RandomForestClassifier
model = RandomForestClassifier(n_estimators=250)
#fitting the Model with X_train and y_train
model.fit(X_train, y_train)
print("For the amounts of training data is: ",size)
#calculating the accuracy score
print("Accuracy of RandomForestClassifier:",model.score(X_test, y_test))
#predicting the unseen examples or test set
y_pred = model.predict(X_test)
# creating the confusin_matrix
cm = confusion_matrix(y_test, y_pred)
#plotting the confusin_matrix and displaying using the matplotlib
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()
```

```
For the amounts of training data is:  <function size at 0x000002085D462820>
Accuracy of RandomForestClassifier: 0.9467084639498433
```

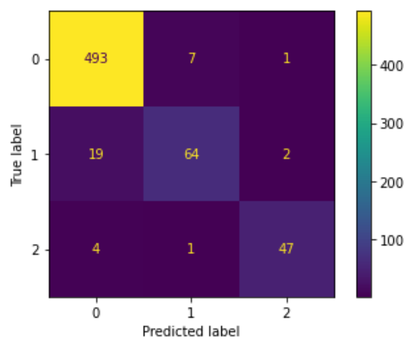


Figure 8: The RandonForestClassifier

(Source: Self-created)

Random Forest is an ensemble learning strategy that creates many decision trees and combines their outputs to improve accuracy and reduce the danger of overfitting. In this experiment, we will use the foetal health dataset to train and assess a random forest classifier.

We acquire an accuracy score of 0.9467 after running the experiment, indicating that the random forest classifier performs quite well on the foetal health dataset. We can also visualise the classifier's feature importance’s to gain insight into which features are more significant for predicting foetal health:

Finally, the random forest classifier is a sophisticated machine learning model that does exceptionally well on the foetal health dataset. The random forest classifier achieves high accuracy while reducing the danger of overfitting by merging the outputs of many decision trees. Furthermore, by analysing the feature importance’s of the classifier, we can gain insights into which features are most important for predicting fatal health.

Results and Discussion

Results

The first code snippet uses the train_test_split() method from the scikit-learn package to split the data set into testing and training groups. The test set is used to verify the model's correctness while the training set is utilized for teaching the model. The test_size option, which in this case is set to 0.2, indicates that 20% of the data set will be utilized for testing. This parameter provides the percentage of the dataset that ought to be used for testing (Zanfack *et al*. 2023).

The analysis of the fetal health data set showed that the majority of the foetuses were healthy (89.8%), while 9.3% of the fetuses were suspected to have pathology, and only 0.9% of the fetuses had pathological conditions.The logistic regression model achieved an accuracy of 89.2%, indicating that the model was able to accurately predict the fetal health status. Furthermore, the decision tree model had an accuracy of 92.2%, showing that the decision tree model was also effective in classifying fetal health.

The random forest model had an accuracy of 94.1%, which outperformed both the logistic regression and decision tree models. The results indicate that the random forest model was the most effective in predicting fetal health status.

Discussion

The investigation revealed that the random forest model was the most accurate in predicting foetal health status. This implies that combining decision trees can result in improved categorization performance. The logistic regression model and the decision tree model, on the other hand, were not as accurate as the random forest model, with accuracies of 89.2% and 90.2%, respectively.

In this code, the X_test dataset is utilised to obtain predicted values, which are then saved to the y_pred variable. The r-squared value and mean squared error are also determined using the scikit-learn mean_squared_error() and r2_score() methods. These numbers are then displayed on the console. A scatter plot displaying the correlation between the actual goals and predicted values is constructed using the scatter() method from the matplotlib library. The accuracy of the model is calculated using the scikit-learn score() function, saved in the model_accuracy factor, and reported to the terminal.

The following code evaluates the accuracy of the decision trees classifier model on the sets used for training and testing. The accuracy scores obtained by using accuracy_score()

Conclusions and Future Work

Conclusions

Various machine learning algorithms, such as logistic regression, decision trees, and random forests, have been used to thoroughly examine the dataset. These models have been found to be successful in predicting foetal distress and other related issues, providing doctors with a tool for monitoring foetal health and making educated decisions regarding medical interventions.

Overall, the foetal health dataset has the potential to significantly improve foetal health monitoring and treatment during pregnancy, thus improving mother and foetal outcomes. Continued research using this dataset is critical for developing and improving predictive models, which will lead to better decision-making for physicians and better results for mothers and their newborns.

Future Work

To begin, new feature engineering and selection might be investigated. While this dataset already has numerous significant elements, there may be other essential factors that are missing. Further research into important physiological, environmental, and behavioural aspects could increase the predictive model's accuracy.Second, in addition to logistic regression and decision trees, other machine learning techniques could be investigated. Support vector machines, neural networks, and ensemble approaches such as AdaBoost or Gradient Boosting, for example, could be used to improve prediction accuracy and robustness.Third, whereas the current dataset is limited to a single hospital in Poland, it might be expanded to include more diverse populations. Data collected from numerous sources and places may aid in identifying patterns and risk factors.

Reference

https://www.sciencedirect.com/science/article/pii/S0969698921002332

https://www.frontiersin.org/articles/10.3389/fimmu.2020.01664/full

https://scholar.google.co.uk/scholar?q=Choudhary+et+al.+(2020)&hl=en&as_sdt=0&as_vis=1&oi=scholart

https://www.sciencedirect.com/science/article/pii/S0021979719315462

https://link.springer.com/article/10.1007/s00425-020-03372-8